Artikel ini telah dipresentasikan dalam *Innovative and Creative Information Technology Conference* (ICITech) dengan tema *"E-Transaction and Power Play"* yang diselenggarakan oleh Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana – Salatiga, pada 24 November 2016.

Penjadwalan Kelas Praktikum Menggunakan Algoritma *Steepest Ascent Hill Climbing*

[1)]Hendra Waskita, [2)]Hindriyanto Dwi Purnomo, [3)]Hendry

[1)]PT. Wahana Cipta Sinatria
Jl. Ambengan Blok A18 No.30,
Ketabang, Genteng, Kota Surabaya, Jawa Timur 60272

[2,3)]Fakultas Teknologi Informasi
Program Studi Teknik Informatika
Universitas Kristen Satya Wacana
Jl. Diponegoro 52-60, Salatiga 50711, Indonesia

[1)] hendrawaskita01@gmail.com, [2)]hindriyanto.purnomo@staff.uksw.edu, [3)]hendry @staff.uksw.edu

Abstrak

Penentuan jadwal praktikum merupakan permasalahan yang sering dihadapi oleh perguruan tinggi. Keterbatasan jumlah kelas seringkali tidak sebanding dengan kegiatan yang dilakukan. Untuk memaksimalkan penggunaan kelas dan juga tenaga pengajar/asisten praktikum, diperlukan system penjadwalan yang baik. Dalam penelitian ini dirancang system penjadwalan praktikum menggunakan metode *steepest ascent hill climbing.* Penelitian ini menggunakan contoh data kelas praktikum di Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana.

Kata kunci : Penjadwalan, masalah penugasan, algoritma *steepest ascent hill climbing*.

1. Pendahuluan

Penjadwalan kelas praktikum merupakan suatu pekerjaan rutin dalam sistem akademik di Perguruan Tinggi pada awal semester perkuliahan. Penjadwalan kelas praktikum digunakan untuk mengatur jadwal mengajar asisten dosen. Penjadwalan bertujuan agar kegiatan dapat lebih terstruktur dan dapat berjalan dengan baik. Namun dalam pelaksanaannya, jadwal terlambat dikeluarkan karena proses penjadwalan membutuhkan waktu yang lama. Ketentuan jadwal dan data pembentuk jadwal yang beragam merupakan faktor penyebab terlambatnya jadwal. Sebagai contoh adalah setiap ajaran baru jumlah mahasiswa semakin banyak sehingga diperlukan tambahan kelas, namun ketersediaan asisten dosen atau pengajar dan ruang yang belum memadai membuat proses penjadwalan menjadi rumit dan membutuhkan waktu yang lama dalam pembuatan jadwal. Hal ini membuat proses belajar mengajar di awal perkuliahan menjadi kurang efektif. Penelitian yang dilakukan Ashita [1] mengenai penjadwalan ujian skripsi dengan menggunakan algoritma *fuzzy multi-attribute decision making* dapat disimpulkan bahwa penjadwalan dengan faktor ketentuan dan faktor pembentuk jadwal yang berbeda-beda akan lebih membutuhkan waktu.

Proses penjadwalan dilakukan dengan memetakan pengajar atau asisten dosen ke dalam kelas-kelas praktikum dengan mempertimbangkan ketentuan yang berlaku. Asisten dosen atau pengajar yang lebih senior dapat mendaftar di

berbagai mata kuliah, sehingga diperlukan batas beban sks mengajar agar pengajar yang baru dapat memperoleh kelas mengajar. Kemungkinan kelas mengajar pada satu mata kuliah berbeda-beda, oleh karena itu diperlukan pembagian kelas yang merata agar beban mengajar dapat terdistribusi secara merata.

Algoritma *steepest ascent hill climbing* adalah metode mencari kenaikan paling tinggi dari keadaan sekitar untuk mencapai solusi tujuan. Algoritma ini menerapkan metode optimasi dengan melakukan pencarian menggunakan nilai *heuristic.* Proses optimasi merupakan proses manipulasi solusi secara berulang-ulang sampai sesuai dengan tujuan yang ingin dicapai. Algoritma *steepest ascent hill climbing* mampu memberikan solusi permasalahan seperti penjadwalan kelas praktikum. Penelitian yang dilakukan Putranto [2] pada penjadwalan mata kuliah dengan menggunakan algortima *steepest ascent hill climbing* diperoleh informasi bahwa algoritma *steepest ascent hill climbing* memiliki keunggulan dimana semua solusi jadwal yang mungkin akan dibangkitkan kemudian akan diperiksa satu persatu, sehingga diperoleh hasil optimal yang diharapkan yaitu tidak ditemukan dua atau lebih proses belajar mengajar pada ruang yang sama serta beban mengajar tidak melampaui batas yang ditentukan. Metode optimasi seperti algoritma *steepest ascent hill climbing* sesuai untuk penelitian menggunakan data ruang sampel dimana terdapat data yang bermacam-macam dengan ketentuan-ketentuan yang perlu diperhatikan. Diharapkan dengan menggunakan metode optimasi dapat memperoleh hasil penjadwalan yang optimal dan jadwal tidak melanggar ketentuan-ketentuan yang berlaku.

2. Tinjauan Pustaka

Metode *steepest ascent hill climbing* merupakan pengembangan dari metode *hill climbing* dimana metode *hill climbing* adalah teknik optimasi matematis yang termasuk dalam kategori teknik pencarian lokal. Untuk menentukan tujuan, metode *hill climbing* memanfaatkan informasi heuristik. Prosedur algoritma *steepest ascent hill climbing* sebagai berikut: (1) Evaluasi keadaan awal (*initial state).* Jika keadaan awal sama dengan tujuan (*goal state*), maka kembali pada *initial state* dan proses berhenti. Jika tidak, maka jadikan *initial state* sebagai *current state*. (2) Mulai dengan *current state* sama dengan *initial state*. (3) Dapatkan semua pewaris (*successor*) dari *current state* yang dapat dijadikan *next state*. (4) Evaluasi seluruh *successor* tersebut dengan fungsi evaluasi dan catat nilai hasil evaluasi. Jika salah satu *successor* tersebut mempunyai nilai yang lebih baik dari *current state*, maka jadikan *successor* dengan nilai paling baik sebagai *new current state*. Lakukan operasi ini terus-menerus sampai *current state* sama dengan *goal state* atau tidak ada perubahan pada *current state*-nya [3][4].

Penelitian menggunakan algoritma *steepest ascent hill climbing* pernah dilakukan. Penelitian yang dilakukan Pasila [5] dengan mengimplementasikan metode *steepest ascent hill climbing* pada mikrokontroler MCS51 untuk robot mobil pencari rute terpendek dapat diperoleh informasi bahwa algortima *steepest ascent hill climbing* dapat memberikan kecerdasan kepada robot untuk memilih rute terpendek dari *map* yang telah ditentukan terlebih dahulu.

Penjadwalan sering digunakan dalam dunia pendidikan untuk mengatur jadwal belajar mengajar. Penjadwalan memiliki arti sebuah koordinasi orang,

kegiatan dan ruang dalam jangka waktu tetentu [6]. Penjadwalan telah banyak dilakukan penelitian dengan kasus yang berbeda-beda. Penelitian yang dilakukan Trisnawati [7] pada penjadwalan kelas dengan menggunakan metode *tabu search* didapatkan informasi bahwa penjadwalan kelas merupakan kegiatan mengalokasikan beberapa komponen yang terdiri dari mata kuliah, ruang dan waktu berdasarkan ketentuan dan syarat yang berlaku. Proses penjadwalan dengan cara manual dengan banyak *variable* yang terkait maka dibutuhkan ketelitian dalam proses penjadwalan mata kuliah, sehingga proses penjadwalan menjadi terlambat. Proses penjadwalan menerapkan *constrain* dan *pinalty* dalam menentukan solusi dapat memperoleh hasil yang optimal.

Penelitian mengenai penjadwalan dengan menggunakan algoritma *steepest ascent hill climbing* dilakukan oleh Putranto [2] pada penjadwalan mata kuliah dan Wijaya [8] pada perancangan dan implementasi aplikasi penjadwalan petugas ibadah gereja diperoleh informasi bahwa jadwal dengan aspek yang bermacam-macam jika dilakukan dengan menggunakan cara manual maka kurang efisien. Ketentuan yang diterapkan seperti dimungkinkan memiliki lebih dari satu jadwal namun tidak pada hari dan jam yang sama dan ketentuan batas pelayanan atau mengajar. Dengan menerapkan algoritma *steepest ascent hill climbing* yang memiliki keunggulan dengan membangkitkan solusi jadwal yang mungkin kemudian diperiksa satu-persatu dapat menghasilkan *output* penjadwalan mendekati hasil optimal yang diinginkan.

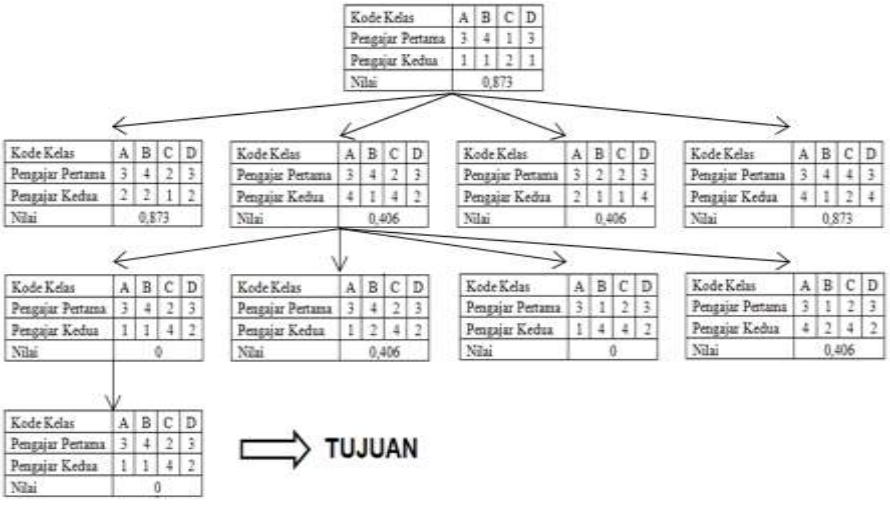
3. Pemodelan Algoritma *Steepest Ascent Hill Climbing* Pada Penjadwalan Kelas Praktikum

Dalam penelitian ini akan dilakukan pemodelan masalah mengenai penjadwalan kelas praktikum dengan menerapkan algoritma *steepest ascent hill climbing*. Adapun perbedaan dari penelitian yang sudah ada adalah pada penelitian yang dilakukan Putranto [2] dan Wijaya [3] solusi jadwal didapatkan dari nilai heuristik yang sudah ditentukan. Jika solusi jadwal yang dibangkitkan tidak sampai pada nilai heuristik yang ditentukan maka jadwal bukan merupakan solusi yang diharapkan, sehingga proses kembali ke proses awal yaitu inisialisasi jadwal dengan cara random. Sedangkan penelitian yang dilakukan adalah jadwal terbaik dari inisialisasi jadwal akan dicek apakah merupakan solusi tujuan. Jika tidak maka jadwal akan dimanipulasi untuk membentuk jadwal baru. Adapun *pseudo code* algoritma *steepest ascent hill climbing* yang akan digambarkan pada Gambar 1.

```
Masuk : Iter Maks, Permasalahan
Keluar : SolusiSekarang
SolusiSekarang ← SolusiRandom(Permasalahan)
Jika (SolusiSekarang == SolusiTujuan)
        Kembali SolusiSekarang
Bukan
        Ulang (Iter i=0 ∈ Iter Maks)
                SolusiKandidat ← DaftarSolusiPewaris(SolusiSekarang)
                Jika (Nilai(SolusiSekarang) > Nilai(SolusiKandidat))
                        SolusiSekarang ← SolusiKandidat
                        i = 0
                        Jika ( SolusiSekarang == SolusiTujuan )
                                Kembali SolusiSekarang
                        Akhir_Jika
                Akhir_Jika
        Akhir_Ulang
Kembali SolusiSekarang
Akhir_Bukan
```

Gambar 1 *Pseudo Code* Algoritma *Steepest Ascent Hill Climbing*

Gambar 1 menggambarkan alur logika program algoritma *steepest ascent hill climbing*. *Pseudo code* pada Gambar 1 dibuat berdasarkan prosedur algoritma *steepest ascent hill climbing.* | SolusiSekarang | merupakan inisialisasi awal, jika merupakan tujuan maka proses berhenti | SolusiSekarang == SolusiTujuan |. Jika tidak maka proses berlanjut dengan membangkitkan solusi yang mungkin | DaftarSolusiPewaris(SolusiSekarang) | berdasarkan solusi sekarang. Jika solusi pewaris | SolusiKandidat | lebih baik dari solusi sekarang | Nilai(SolusiSekarang) > Nilai(SolusiKandidat) |, maka solusi sekarang adalah solusi yang terbaik atau solusi pewaris | SolusiSekarang ← SolusiKandidat |. Proses akan berlansung sampai solusi sekarang merupakan solusi tujuan atau solusi sekarang tidak mengalami perubahan sebanyak jumlah yang ditentukan | ***Iter*** *Maks* |. Adapun ilustrasi dari prosedur algoritma *steepest ascent hill climbing* akan digambarkan pada Gambar 2.



Gambar 2 Ilustrasi Alur Algoritma *Steepest Ascent Hill Climbing*

Gambar 2 menggambarkan prosedur dari inisialisasi jadwal sampai menemukan solusi tujuan. Pada penelitian ini jadwal tujuan adalah jadwal yang tidak melanggar *constrain* sehingga jadwal memiliki nilai 0. Pada Gambar 2, inisialisasi jadwal memiliki nilai 0,873. Karena jadwal bukan merupakan tujuan, kemudian jadwal dimanipulasi dengan mengubah struktur jadwal, sehingga didapatkan jadwal yang lebih baik seperti jadwal dengan nilai 0,406. Jadwal bukan merupakan jadwal tujuan, sehingga jadwal akan dimanipulasi ke proses yang kedua. Proses manipuasi kedua sama dengan proses manipuasi pertama, kemudian dihasilkan jadwal yang terbaik adalah jadwal bernilai 0. Jadwal bernilai 0 sesuai dengan nilai jadwal tujuan sehingga proses berhenti.

Pada Gambar 2 terdapat jadwal dengan nilai yang sama dengan jadwal sebelumnya. Berdasarkan prosedur algoritma *steepest ascent hill climbing*, jika jadwal pewaris lebih baik maka jadwal sekarang akan diubah dengan jadwal pewaris. Sehingga jadwal yang bernilai sama bukan jadwal yang lebih baik dari jadwal sebelumnya. Berdasarkan hal tersebut, maka jadwal sebelumnya tidak akan diubah dengan jadwal pewaris.

Membangkitkan jadwal baru atau proses manipulasi berdasarkan cek *constrain* pada setiap pengajar. jika pengajar yang melanggar *constrain*, maka pengajar akan dirandom untuk mendapatkan kelas baru. Jika pengajar tidak melanggar *constrain*, maka kelas yang didapatkan tetap. Dari proses tersebut terbentuk jadwal baru yang merupakan turunan dari jadwal sebelumnya, dimana masih terdapat kesamaan dari jadwal sebelumnya. Selanjutnya evaluasi jadwal, jika jadwal baru lebih baik maka proses berikutnya menggunakan jadwal baru. Jika tidak maka proses selanjutnya menggunakan jadwal sebelumnya. Proses manipulasi jadwal akan dikerjakan sampai jadwal sudah sesuai dengan tujuan atau jadwal tidak berubah sampai batas yang sudah ditentukan.

Dalam penelitian yang dilakukan terdapat dua *constrain* yaitu *hard constrain* dan *soft constrain. Hard constrain* merupakan ketentuan yang tidak boleh dilanggar dan *soft constrain* merupakan ketentuan yang memiliki toleransi untuk dilanggar, namun tetap diharapkan tidak melanggar ketentuan tersebut. *Hard constrain* jadwal adalah pengajar tidak boleh mengajar pada hari dan jam yang sama di kelas yang berbeda. Kelas pada hari dan jam yang sama tidak boleh digunakan untuk lebih dari satu mata kuliah. Setiap kelas harus terdapat kandidat pengajar. Pada penelitian ini *hard constrain* dicek pada saat akan memulai membuat jadwal. Sedangkan *soft constrain* jadwal adalah pengajar tidak boleh melebihi batas beban sks pengajar dan setiap pengajar mendapatkan jumlah kelas repatif sama sesuai dengan jumlah rata-rata kelas di dalam jadwal. Dua *soft constrain* tersebut merupakan kriteria untuk menentukan kualitas jadwal. Rumus untuk menghitung kualitas jadwal $|f_{(ob)}|$ akan dijelaskan sebagai berikut :

$$f_{(ob)} = \frac{C_1 \text{ x } f_{(sks)} + C_2 \text{ x } f_{(kelas)}}{C_1 + C_2} \quad (1)$$

$$f_{(sks)} = \frac{\sum^{n}_{i=1} sks_i}{n} \quad (2)$$

$$\left\{ \text{jika } jsks_{(s+b)} < maks_{(sks)} = 0 \right\}$$

$$sks_i = \text{atau} \quad jsks_{(s+b)} > maks_{(sks)} = jsks_{(s+b)} - maks_{(sks)} \tag{3}$$

$$f_{(kelas)} = \sqrt{\frac{\sum^{n}_{i=1} (\overline{x}_i - x)^2}{n-1}} \tag{4}$$

$$\overline{x} = \frac{\sum^{n}_{i=1} x_i}{n} \tag{5}$$

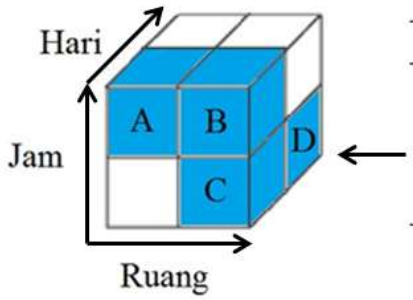
Keterangan :

$f_{(ob)}$ = Fungsi objektif jadwal
$f_{(sks)}$ = Fungsi beban sks mengajar
$f_{(kelas)}$ = Fungsi penyetaraan kelas
C_1, C_2 = Nilai penyetaraan
sks_i = Fungsi pinalti beban sks tiap pengajar
n = Jumlah pengajar
$jsks_{(s+b)}$ = Jumlah sks jadwal yang sudah dibuat dan akan dibuat
$maks_{(sks)}$ = Batas maksimal sks
x_i = Jumlah kelas pengajar
$\overline{x}$ = Rata-rata kelas

Fungsi objektif jadwal | $f_{(ob)}$ | terdiri dari penjumlahan fungsi beban sks | $f_{(sks)}$ | dan fungsi penyetaraan kelas | $f_{(kelas)}$ | dimana nilai | $f_{(sks)}$ | dan nilai | $f_{(kelas)}$ | disetarakan dengan notasi | C_1 | dan | C_2 |. Penyetaraan dilakukan karena nilai dari | $f_{(sks)}$ | dan | $f_{(kelas)}$ | tidak sebanding. Rumus fungsi beban sks mengajar terdapat notasi | $\sum sks_i$ | yang juga merupakan rumus pinalti atau batas beban sks setiap pengajar. Jika jumlah beban sks mengajar yang didapatkan | $jsks_{(s+b)}$ | kurang dari ketentuan | $maks_{(sks)}$ | maka nilai beban sks mengajar adalah nol. Jika jumlah beban sks mengajar melebihi ketentuan maka nilai yang didapatkan adalah jumlah beban sks mengajar yang sudah didapatkan dikurangi jumlah beban sks mengajar yang ditentukan | $jsks_{(s+b)} - maks_{(sks)}$ |. Setelah mendapatkan nilai beban sks mengajar pada setiap pengajar selanjutnya akan dihitung rata-rata nilai beban sks mengajar pada setiap pengajar | $f_{(sks)}$ |.

Fungsi pemerataan kelas | $f_{(kelas)}$ | menggunakan rumus standar deviasi karena jumlah kelas setiap pengajar yang berbeda-beda. Jika seorang pengajar mendapatkan kelas sesuai dengan rata-rata jumlah pembagian kelas, maka nilai adalah nol. Jika kurang atau melebihi rata-rata maka akan memiliki suatu nilai. Jika nilai beban sks jadwal dan nilai kesetaraan kelas sudah didapat maka nilai kualitas jadwal dapat diketahui dengan fungsi objektif jadwal. Selanjutnya akan menjelaskan tentang pemetaan dalam proses membuat jadwal. Pemetaan jadwal menggunakan faktor kelas praktikum yang memiliki parameter hari, jam dan ruang. Faktor pengajar dengan parameter identitas pengajar dan data kemungkinan kelas mengajar. Proses membuat jadwal yang dilakukan adalah dengan membagi pengajar ke dalam kelas praktikum. Diasumsikan bahwa data pengajar sudah didapatkan dari proses seleksi asisten dosen atau pengajar yang sebelumnya sudah dilakukan. Untuk data kelas praktikum merupakan data kelas

yang didapatkan dari bagian Tata Usaha (TU), sehingga data kelas praktikum diharapkan tidak mengalami bentrok antar kelas. Proses membuat jadwal akan dipetakan pada Gambar 3.



Pengajar/Kelas	A	B	C	D
Pengajar 1	V	V	V	V
Pengajar 2	V	V	V	V
Pengajar 3	V	V	V	V
Pengajar 4	V	V	V	V

Gambar 3 Pemetaan Jadwal

Gambar 3 menjelaskan pemetaan jadwal dengan mengalokasikan pengajar pada kelas yang sudah tersedia, berdasarkan kemungkinan kelas mengajar yang sudah didaftarkan oleh pengajar. Data kelas digambarkan dengan matrik tiga dimensi yang diasumsikan bahwa dalam satu ruang dapat digunakan pada hari yang sama dengan jam yang berbeda. Dalam proses algoritma *steepest ascent hill climbing* terdapat proses manipulasi jadwal baru yang merupakan turunan dari jadwal yang sebelumnya. Pemetaan proses manipulasi jadwal baru akan digambarkan pada Gambar 4.

Jadwal Random

Kelas\Pengajar	Pengajar 1	Pengajar 2
A	1	3
B	4	1
C	1	3
D	2	4

Jadwal Awal

Pengajar 1	Pengajar 2	Pengajar 3	Pengajar 4
A , B , C	D	A , C	B , D

Optimasi

RANDOM		TETAP	
Pengajar	1 , 2	3	4
Kelas	A , B , C , D	A , C	B , D

Jadwal Akhir

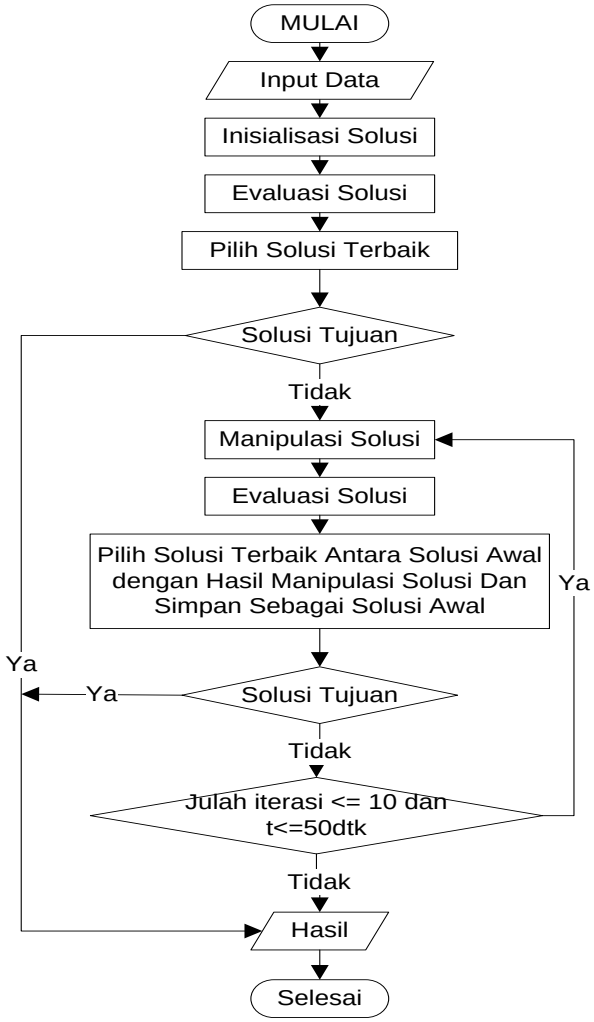
Pengajar 1	Pengajar 2	Pengajar 3	Pengajar 4
A , B	C , D	A , C	B , D

Gambar 4 Pemetaan Manipulasi Jadwal

Gambar 4 menjelaskan gambaran sistem dalam mendeteksi pengajar yang melanggar *soft constrain*. Jika terdapat pengajar yang melanggar *soft constrain* maka kelas yang diterima akan dilakukan random dengan kandidat pengajar yang sama-sama melanggar *soft constrain*. Jika kelas yang dirandom tidak sesuai dengan kemungkinan kelas mengajar pada setiap kandidat pengajar atau jika kandidat pengajar hanya satu pengajar maka kandidat pengajar akan ditambah.

Penambahan kandidat pengajar bertujuan untuk menciptakan kemungkinan jadwal baru. Untuk pengajar yang tidak melanggar *soft constrain*, maka kelas tetap. Hasil random akan digabungkan dengan kelas yang sudah tetap, sehingga membentuk jadwal baru.

Jika jumlah kandidat pengajar dengan jumlah kelas tidak dapat menemukan nilai rata-rata berupa angka bilangan bulat maka pemerataan kelas tidak akan tercapai dan jika dalam satu kelas sangat sedikit kandidat pengajar maka dimungkinkan pengajar akan mendapatkan beban sks mengajar yang melebihi batas maksimal beban sks. Jika dalam membuat jadwal terdapat jadwal yang melanggar *soft constrain* maka jadwal akan tetap ditampilkan, hal ini bertujuan untuk memberikan contoh jadwal. Proses berikutnya adalah perancangan algoritma yang diterapkan di dalam sistem berdasarkan prosedur dalam algoritma *steepest ascent hill climbing*. Perancangan algoritma yang digambarkan dengan *flowchart diagram* pada Gambar 5.



Gambar 5 *Flowchart* Algoritma *Steepest Ascent Hill Climbing*

Gambar 5 menjelaskan tahapan mulai sampai tahapan selesai dari alur program yang mengimplementasikan algoritma *steepest ascent hill climbing*.

Tahap *input* data adalah pembuat jadwal melakukan *input* data berupa tahun, semester, nama mata kuliah dan maksimal sks. Data ini digunakan untuk menentukan data mata kuliah, data praktikum, data ruang, data pengajar dan ketentuan jadwal. Tahap inisialisasi solusi adalah membangkitkan solusi yang mungkin sebagai solusi tujuan. Tahap evaluasi solusi adalah memberikan nilai pada setiap solusi jadwal. Tahapan pilih solusi terbaik adalah memilih jadwal terbaik berdasarkan nilai jadwal. Solusi jadwal yang mendapat nilai kecil merupakan jadwal yang melanggar *soft constrain* paling sedikit dan juga sebaliknya. Tahap cek solusi tujuan adalah melakukan cek pada solusi jadwal dengan membandingkan nilai solusi jadwal dengan nilai jadwal tujuan. Jika nilai dari solusi jadwal sama dengan nilai solusi jadwal tujuan maka proses berhenti. Jika tidak maka akan dilanjutkan proses berikutnya.

Tahap manipulasi jadwal adalah proses membuat jadwal baru dengan cara mengubah struktur jadwal berdasarkan ketentuan yang ada di dalam sistem. Solusi jadwal yang didapatkan dari tahap cek solusi terbaik akan diinisialisasi sebagai jadwal awal dan hasil dari manipulasi jadwal adalah jadwal baru. Setelah cek solusi tujuan selanjutnya dilakukan cek jumlah iterasi. Tahap cek jumlah iterasi adalah cek jadwal tidak berubah sebanyak 10 kali perbandingan dengan jadwal baru. Jika jadwal tidak berubah sebanyak 10 kali perbandingan dan waktu proses kurang atau sama dengan 50 detik maka proses berhenti dan jadwal akan ditampilkan. Jika tidak maka proses berlanjut ke tahap ketujuh atau manipulasi jadwal. Tahap selesai akan menampilkan hasil berupa jadwal dari *generate* jadwal.

4. Hasil dan Pembahasan

Algoritma *steepest ascent hill climbing* untuk penjadwalan kelas praktikum diimplementasikan pada bahasa pemograman PHP dan HTML5. Desain tampilan menggunakan CSS dan teknologi *javascript*. Basis data yang digunakan adalah basis data MySql pada aplikasi wampserver 5.1.13. Untuk *hardware* menggunakan prosessor i3 2,4GHz dan RAM 4 GB. Dengan spesifikasi tersebut aplikasi yang dibuat dapat berjalan dengan baik.

Aturan yang digunakan untuk menguji sistem adalah dari Fakultas Teknologi Informasi Universitas Kristen Satya Wacana (UKSW) Salatiga. Untuk data kelas praktikum dan data pengajar menggunakan data sampel untuk menguji sistem. Dalam pembahasan ini, informasi mata kuliah yang akan digunakan untuk uji sistem akan dijelaskan pada Tabel 1.

Tabel 1 Keterangan Mata Kuliah Pada Uji Sistem Penjadwalan

	Keterangan
Mata Kuliah	Pemrograman berorientasi objek dan kode mata kuliah IT300 dengan beban sks bayar 3/4.
Praktikum	Kode praktikum IT300A dengan jumlah kelas 15 kelas. Beban sks mengajar 2 sks/kelas.
Ruang	Kode ruang E201A dan RX302 dengan jumlah pengajar 2 orang/kelas.

Pengajar	Jumlah asisten dosen 15 orang. Beban maskimal sks mengajar adalah 18 sks untuk semua pengajar.
Uji Sistem	Beban sks pengajar 1 sejumlah 16 sks. Kelas D dan H memiliki hari dan jam yang sama dengan ruang yang berbeda. Kelas B dan L memiliki hari dan jam yang sama dengan ruang yang berbeda.

Tabel 1 menjelaskan mengenai informasi dan aturan mata kuliah yang akan diuji coba dilakukan penjadwalan dan beberapa kondisi untuk menguji sistem. Pengujian performa motode *steepest ascent hill climbing* dilakukan dengan membandingkan dengan metode random. Setiap metode dijalankan sebanyak 30 kali. Setiap satu kali menjalankan metode, jumlah evaluasi | $f_{(ob)}$ | sebanyak 25 kali. Batas waktu satu kali menjalankan metode adalah 50 detik. Dengan pengujian metode ini dapat disimpulkan metode mana yang lebih baik. Selanjutnya proses membuat jadwal, dengan data kelas dan data kemungkinan mengajar. Data kelas praktikum dijelaskan pada Tabel 2.

Tabel 2 Data Kelas Praktikum

MATAKULIAH PRAKTEK		HARI	JAM	KAPASITAS	RUANG
Pemrograman Berbasis Objek	A	Selasa	10-12	40	RX302
Pemrograman Berbasis Objek	B	Selasa	15-17	40	RX302
Pemrograman Berbasis Objek	C	Rabu	17-19	40	RX302
Pemrograman Berbasis Objek	D	Jumat	16-18	40	RX302
Pemrograman Berbasis Objek	E	Kamis	18-20	40	RX302
Pemrograman Berbasis Objek	F	Jumat	07-09	40	RX302
Pemrograman Berbasis Objek	G	Jumat	14-16	40	RX302
Pemrograman Berbasis Objek	H	Jumat	16-18	40	E201A
Pemrograman Berbasis Objek	I	Jumat	18-20	40	RX302
Pemrograman Berbasis Objek	J	Sabtu	07-09	40	RX302
Pemrograman Berbasis Objek	K	Sabtu	09-11	40	RX302
Pemrograman Berbasis Objek	L	Selasa	14-16	40	RX302
Pemrograman Berbasis Objek	M	Sabtu	07-09	40	E201A
Pemrograman Berbasis Objek	N	Sabtu	09-11	40	E201A
Pemrograman Berbasis Objek	O	Rabu	12-15	40	RX302

Tabel 2 merupakan daftar kelas praktikum pada mata kuliah pemrograman berbasis objek. Data kelas pada Tabel 2 digunakan untuk pengujian algoritma karena terdapat dua kelas dengan hari dan jam yang sama seperti kelas D dan H. Data kemungkinan kelas mengajar asisten dosen atau pengajar ditunjukan pada Tabel 3.

Tabel 3 Data Kelas Mengajar Asisten Dosen

Pengajar/kelas	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Pengajar 1	V	V	V	V	V	V	V	V							
Pengajar 2									V	V	V	V	V	V	V
Pengajar 3				V	V	V				V	V	V			
Pengajar 4	V	V	V				V	V	V				V	V	V
Pengajar 5			V	V			V	V			V	V			V
Pengajar 6	V	V			V	V			V	V			V	V	
Pengajar 7		V		V		V		V		V		V		V	
Pengajar 8	V		V		V		V		V		V		V		V
Pengajar 9		V			V	V				V	V	V			
Pengajar 10	V		V	V			V	V	V				V	V	V
Pengajar 11	V	V	V	V	V										
Pengajar 12						V	V	V	V	V					
Pengajar 13											V	V	V	V	V
Pengajar 14			V	V	V	V	V	V	V	V	V	V	V	V	V
Pengajar 15	V	V													

Tabel 3 adalah menjelaskan kemungkinan kelas mengajar, jika terdapat tanda “V” maka pengajar dapat mengajar di kelas tersebut dan jika tidak maka pengajar tidak dapat mengajar di kelas tersebut. Berdasarkan proses algoritma *steepest ascent hill climbing* akan dibangkitkan jadwal yang mungkin sebagai tujuan. Jadwal akan dibangkitkan secara random sebagai inisialisasi. Jumlah jadwal inisialisasi sebanyak kemungkinan kombinasi yang dapat dilakukan, jika jumlah terlalu banyak maka sistem akan membatasi dengan cara inisialisasi jadwal akan diberikan waktu selama 10 detik dan minimal jadwal yang dibentuk sebanyak dua jadwal, sehingga jika sudah mencapai 10 detik namun inisialisasi jadwal kurang dari dua maka proses inisialisasi tetap dilakukan. Dalam proses random, sistem akan melakukan cek mengajar. Cek mengajar adalah proses cek pada jumlah kemungkinan mengajar, jika kemungkinan kelas mengajar kurang atau sama dengan jumlah rata-rata pembagian kelas maka pengajar tersebut akan diprioritaskan untuk mendapatkan kelas terlebih dahulu. Dalam melakukan random, sistem juga melakukan cek pada kelas yang memiliki jam dan hari yang sama ditempat yang berbeda. jika salah satu pengajar mendaftarkan kelas yang memiliki hari dan jam yang sama, maka sistem akan menempatkan pengajar disalah satu kelas. Sebagai contoh jadwal random dijelaskan pada Tabel 4.

Tabel 4 Jadwal Random

Pengajar/Kelas	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Pengajar	15	15	11	12	12	13	10	4	3	3	1	14	2	12	2
Pengajar	13	3	6	14	5	8	9	10	4	14	7	4	7	11	13

Tabel 4 menjelaskan mengenai jadwal random yang sudah dibuat berdasarkan ketentuan pada Tabel 1. Pada Tabel 1, pengajar 15 mendaftarkan jumlah kelas mengajar sama dengan jumlah rata-rata kelas yaitu dua kelas

sehingga pada Tabel 3 pengajar 15 juga sudah mendapat dua kelas mengajar. Pada pengajar 3, 12, 14 dan 15 tidak terjadi bentrok kelas pada kelas D dan H dan pada pengajar 4, 10, 11 dan 12 tidak terjadi bentrok kelas pada kelas B dan L. Selanjutnya jadwal akan dihitung kualitas jadwal dan dijelaskan pada Tabel 5.

Tabel 5 Daftar Nilai Jadwal Random

Pengajar	Jml Kelas	Jml Sks	Jml Sks Db	Total Sks
Pengajar 1	1	2	16	18
Pengajar 2	2	4	0	4
Pengajar 3	3	6	0	6
Pengajar 4	3	6	0	6
Pengajar 5	1	2	0	2
Pengajar 6	1	2	0	2
Pengajar 7	2	4	0	4
Pengajar 8	1	2	0	2
Pengajar 9	1	2	0	2
Pengajar 10	2	4	0	4
Pengajar 11	2	4	0	4
Pengajar 12	3	6	0	6
Pengajar 13	3	6	0	6
Pengajar 14	3	6	0	6
Pengajar 15	2	4	0	4

Tabel 5 menjelaskan nilai yang didapatkan dari setiap pengajar seperti jumlah kelas, jumlah sks mengajar pada jadwal pemrograman berorientasi objek , jumlah beban sks yang sudah didapatkan di jadwal lain dan total sks yang dimiliki pengajar. Berdasarkan Tabel 5 dapat disimpulkan bahwa inisialisasi jadwal dengan cara random tidak sesuai dengan ketentuan penjadwalan. Terdapat pengajar yang tidak sesuai dengan pemerataan kelas seperti pada pengajar 1, 3, 4, 5, 6, 8, 9, 12, 13 dan 14. Untuk ketentuan batas beban sks tidak terjadi pelanggaran atau tidak ada yang melebihi batas 18 sks. Berdasarkan cek ketentuan, selanjutnya jadwal akan dinilai menggunakan rumus yang sudah dijelaskan sebelumnya. Perhitungan jadwal menggunakan rumus kesetaraan nilai memiliki notasi nilai | C_1 | bernilai 3,5 dan | C_2 | bernilai 16,5. Rumus pinalti memiliki notasi nilai | $Maks_{(sks)}$ | bernilai 18 dan | n | bernilai 15. Contoh perhitungan akan dijelaskan sebagai berikut.

$$f_{(ob)} = \frac{(3{,}5 \text{ x } f_{(kelas)})+(16{,}5 \text{ x } f_{(sks)})}{(3{,}5 + 16{,}5)} = 0{,}127$$

$$= \frac{(3{,}5\text{x}0{,}845)+(16{,}5\text{x}0{,}000)}{20} = 0{,}127$$

$$f_{(sks)} = \frac{(0 + 0 + 0 + 0 + 0 +...+ 0)}{15} = 0{,}000$$

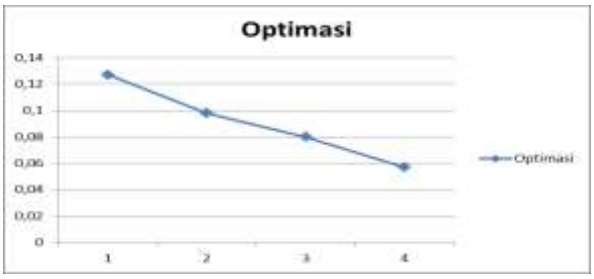
$$f_{(kelas)} = \sqrt{\frac{(1-2)^2+(2-2)^2+\ldots+(2-2)^2}{(15-1)}} = 0{,}845$$

$$\overline{x} = \frac{(1+2+3+3+1+1+2+1+\ldots+2)}{15} = 2$$

Perhitungan nilai kualitas jadwal didapat berdasarkan Tabel 5. Berdasarkan perhitungan tersebut, nilai kualitas jadwal dari hasil random adalah 0,127 karena terjadi pelanggaran *soft constrain*. Nilai beban sks jadwal $|f_{(sks)}|$ adalah 0,000 yang berarti bahwa tidak ada pengajar yang melanggar batas 18 sks seperti yang dijelaskan pada Table 5. Nilai kesetaraan kelas $|f_{(kelas)}|$ adalah 0,845 yang berarti kesetaraan kelas tidak terpenuhi. Selanjutnya adalah proses evaluasi jadwal dan proses manipulasi jadwal. Perubahan nilai pada proses manipulasi jadwal akan dijelaskan pada Tabel 6 dan Gambar 6.

Tabel 6 Daftar Nilai Proses Manipulasi Jadwal

Jml Manipulasi	$f_{(kelas)}$	$f_{(sks)}$	$f_{(ob)}$
1	0,845	0,000	0,127
2	0,655	0,000	0,098
3	0,535	0,000	0,08
4	0,378	0,000	0,057



Gambar 6 Grafik Proses Manipulasi Jadwal

Tabel 6 merupakan daftar nilai perubahan jadwal dan Gambar 6 menjelaskan grafik perubahan nilai jadwal. Karena nilai jadwal manipulasi tidak menemui tujuan atau nilai jadwal nol maka proses berlanjut sampai jadwal tidak mengalami perubahan samapi batas yang ditentukan. Dalam proses manipulasi jadwal terjadi tiga kali perubahan jadwal. Jadwal akan dijelaskan pada Tabel 7.

Tabel 7 Hasil Manipulasi Jadwal

Pengajar/Kelas	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**
Pengajar	15	15	11	9	12	8	10	10	3	13	1	3	2	11	8
Pengajar	13	4	4	13	6	5	6	14	5	12	7	14	7	9	2

Tabel 7 merupakan jadwal hasil dari proses manipulasi. Nilai kesetaraan kelas jadwal adalah 0,378 yang berarti terdapat pengajar yang mendapat jumlah

kelas tidak sesuai dengan jumlah rata-rata pembagian kelas. Nilai batas beban sks adalah 0,000 yang berarti tidak ada pengajar yang mendapatkan beban sks melebihi batas ketentuan yaitu 18 sks. Nilai jadwal yang didapatkan adalah 0,057. Jadwal sudah optimal karena pengajar tidak melebihi batas sks namun terdapat pengajar yang mendapat jumlah kelas tidak sesuai jumlah rata-rata kelas. Hal ini dikarenakan pengajar sudah mendapatkan kelas mengajar di jadwal lain. Tabel 7 juga menjelaskan bahwa manipulai jadwal baru merupakan jadwal yang dibentuk dari jadwal sebelumnya, seperti kotak berlatar belakang putih masih tetap sama dengan jadwal pada Tabel 4. Selanjutnya akan dilakukan uji metode yang bertujuan untuk mengetahui apakah dengan menggunakan algoritma dapat memperoleh hasil lebih baik dibandingan dengan menggunakan cara random. Proses pengujian akan digambarkan pada Tabel 8.

Tabel 8 Uji Metode

	Steepest Ascent Hill Climbing				**Random**		
NO	$f_{(kelas)}$	$f_{(sks)}$	$f_{(ob)}$	**NO**	$f_{(kelas)}$	$f_{(sks)}$	$f_{(ob)}$
1	0,378	0,000	0,057	1	0,845	0,000	0,127
2	0,655	0,000	0,098	2	1,069	0,000	0,16
3	0,378	0,000	0,057	3	0,926	0,000	0,139
4	0,378	0,000	0,057	4	0,655	0,000	0,098
5	0,378	0,000	0,057	5	0,845	0,000	0,127
6	0,378	0,000	0,057	6	0,756	0,000	0,113
7	0,378	0,000	0,057	7	0,756	0,000	0,113
8	0,378	0,000	0,057	8	0,756	0,000	0,113
9	0,378	0,000	0,057	9	0,845	0,000	0,127
10	0,378	0,000	0,057	10	0,655	0,000	0,098
11	0,535	0,000	0,08	11	0,756	0,000	0,113
12	0,378	0,000	0,057	12	0,756	0,000	0,113
13	0,378	0,000	0,057	13	0,756	0,000	0,113
14	0,535	0,000	0,08	14	0,926	0,000	0,139
15	0,378	0,000	0,057	15	0,756	0,000	0,113
16	0,378	0,000	0,057	16	0,756	0,000	0,113
17	0,378	0,000	0,057	17	0,926	0,000	0,139
18	0,378	0,000	0,057	18	0,845	0,000	0,127
19	0,378	0,000	0,057	19	0,756	0,000	0,113
20	0,378	0,000	0,057	20	0,655	0,000	0,098
21	0,378	0,000	0,057	21	0,756	0,000	0,113
22	0,378	0,000	0,057	22	0,655	0,000	0,098
23	0,378	0,000	0,057	23	0,756	0,000	0,113
24	0,378	0,000	0,057	24	0,756	0,000	0,113
25	0,535	0,000	0,08	25	0,845	0,000	0,127
26	0,378	0,000	0,057	26	0,655	0,000	0,098
27	0,378	0,000	0,057	27	0,655	0,000	0,098
28	0,378	0,000	0,057	28	0,756	0,000	0,113
29	0,378	0,000	0,057	29	0,756	0,000	0,113
30	0,378	0,000	0,057	30	0,655	0,000	0,098

Tabel 8 menjelaskan bahwa, dalam menggunakan metode *steepest ascent hill climbing* dan metode random dapat meminimalkan pelanggaran batas sks | $f_{(sks)}$ |. Hal tersebut dibuktikan dari kedua metode pada Tabel 8 nilai batas sks | $f_{(sks)}$ | adalah 0,000. Hasil keseluruhan dari uji metode ditunjukan pada Tabel 9.

Tabel 9 Hasil Uji Metode

	Steepest Ascent Hill Climbin	**Random**
S.DEV	0,009	0,019
MEAN	0,06	0,113

Tabel 9 menunjukan perhitungan dalam uji metode pada Tabel 8. Berdasarkan hasil pada Tabel 9 diperoleh hasil bahwa metode *steepest ascent hill climbing* memiliki nilai *standar deviasi* adalah 0,009 dan nilai *mean* adalah 0,06. Sedangkan uji dengan metode random memiiki nilai *standar deviasi* adalah 0,019 dan nilai *mean* adalah 0,0113. Berdasarkan nilai tersebut dapat disimpulkan bahwa penjadwalan menggunakan metode algoritma *steepest ascent hill climbing* lebih baik dari pada menggunakan metode cara random.

5. Simpulan

Berdasarkan penelitian yang sudah dibuat dengan judul penerapan algoritma *steepest ascent hill climbing* pada penjadwalan kelas praktikum diperoleh beberapa kesimpulan . Pada penelitian ini algoritma dapat diterapkan dengan baik dan dapat menghasilkan solusi sesuai dengan tujuan yang diharapkan atau mendekati solusi tujuan. Banyaknya aturan penjadwalan akan membuat proses penjadwalan menjadi rumit namun jika data yang digunakan banyak akan mempermudah proses algoritma dalam menentukan solusi. Berdasarkan pengujian yang dilakukan, metode algoritma *steepest ascent hill climbing* lebih efisien sebesar 35,72% dari pada menggunakan metode random. Penjadwalan mengguakan sistem penjadwalan secara terkomputerisasi akan lebih mudah dan lebih cepat. Penjadwalan disimpan dalam *database* sehingga memudahkan dalam *backup* data. Data saling terintegrasi sehingga meminimalkan terjadinya seorang pengajar mengajar pada hari dan jam yang sama di mata kuliah yang berbeda.

Beberapa saran untuk pengembangan penelitian ini adalah aplikasi dapat melakukan penjadwalan lebih dari satu mata kuliah dengan proses penjadwalan yang lebih singkat. Sistem yang dibangun dapat berintegrasi dengan jadwal perkuliahan asisten dosen sehingga data kemungkinan mengajar akan lebih tepat.

6. Pustaka

[1] Ashita, Aghata Dhiwi., Martin Setyawan, dan Yessica, Penerapan Algoritma *Fuzzy Multi-Attribute Decision Making* pada Penjadwalan Ujian Skripsi (Studi Kasus : Fakultas Teknologi Informasi – Universitas Kristen Satya Wacana Salatiga) Salatiga : Fakultas Teknologi Informasi Universitas Kristen Satya Wacana Salatiga.

[2] Putranto, Klaudius Nikotino., M. A. Ineke Pakereng dan Ramos Somya, 2012, Perancangan dan Implementasi Penjadwalan ata Kuliah

Menggunakan Algoritma *Steepest Ascent Hill Climbing* (Studi Kasus : Fakulas Psikologi UKSW), Salatiga: Fakultas Teknologi Informasi Universitas Kristen Satya Wacana Salatiga

[3] Kusumadewi, Sri., 2003, *Artificial Intelligence (Teknik dan Aplikasinya)*, Yogyakarta : Graha Ilmu.

[4] http://www.cse.msstate.edu/~bridges/ai/lecture4/sld011.htm, Diakses 11 September 2014.

[5] Pasila, Felix., Thiang, Handry Khoswanto, Ferdi Ninaber, dkk, 2009, Implementasi Metode *Steepest Ascent Hill Climbing* pada Mikrokontroler MCS51 untuk Robot Mobil Pencari Rute Terpendek, Surabaya : Jurusan Teknik Elektro Universitas Kristen Petra.

[6] Carleton University, 2007, Coordinated Timetabling : Principles, Rules & Responsibilities.

[7] Trisnawati, Ade., Iriansyah BM Sangadji dan Sely Karmila, 2011, Implementasi Metode Tabu Search untuk Penjadwalan Kelas, Jakarta Barat : Teknik Informatika Sekolah Tinggi Teknik PLN.

[8] Wijaya, Candra., 2013, Perancangan dan Implementasi Aplikasi Penjadwalan Petugas Ibadah Gereja Menggunakan Algoritma *Steepest Ascent Hill Climbing* (Studi Kasus : Gereja Mawar Sharon Kemah Kemenangan Salatiga), Salatiga : Fakultas Teknologi Informasi Universitas Kristen Satya Wacana Salatiga.

[9] Pressman, Roger S., 2001, *Rekayasa Perangkat Lunak Pendekatan Praktisi (Buku Satu)*, Yogyakarta : Andi.

[10] Dewi, Muhammad Rezki Firdaus dan Syarif Indra Halim., 2009, Penerapan Metode *Hill Climbing Search* Untuk Pencarian Lokasi Terdekat Pada Aplikasi Toko Virtual Berbasis Android : Jurusan Teknik Informatika STMIK GI MDP.

[11] Fathansyah, 1999, *Basis Data*, Informatika Bandung : Bandung.

[12] Hanindito, Gregorius Anung., 2014, Sistem Informasi Pemesanan Tiket Pesawat Secara Online, Salatiga : Fakultas Teknologi Informasi Universitas Kristen Satya Wacana Salatiga.

[13] Hasibuan, Zainal A., 2007, Metodologi Penelitian Pada Bidang Ilmu Komputer dan Teknologi Informasi : Konsep, Teknik, dan Aplikasi, Jakarta : Ilmu Komputer Universitas Indonesia.

[14] Hayder, Hasin., 2007, Object-Oriented Programming with PHP5 : BIRMINGHAM - MUMBAI.

[15] Marlinda, Linda., 2004, *Sistem Basis Data*, Yogyakarta : Andi.

[16] iRich, Elaine., 1991, *Artificial Intelligence*, New York: McGraw-Hill.